3-D Example

PROC IML;
sigma = {1 .2 2,
    .2 1 2,
    2 2 10};
mu = {0 0 0};
n = 1000;
seed = 91505;
q = NROW(sigma);
MUMAT = REPEAT(mu, n, 1);
SROOT = ROOT(sigma);
Z = NORMAL(REPEAT(seed, n, q));
x = Z * SROOT + MUMAT;
CREATE mvnormdata FROM x;
APPEND FROM x;
QUIT;

Goal

Find the coefficients (a’s) of the x’s so that:

\[ Y = a_1X_1 + a_2X_2 + \cdots + a_qX_q \]

has the largest possible variance subject to the condition that the length of the coefficient vector is 1.
Example Cont.

\begin{verbatim}
library(MASS)
mui<-c(5,0,-1)
sigma<-
   matrix(c(1,0.2,2,0.2,1,2,2,2,10),
   ncol=3,byrow=T)
x<-mvrnorm(n=1000,mui,sigma)
coef<-princomp(x,cor=F)$loadings[,1]
pc1<-princomp(x,cor=F)$scores[,1]
\end{verbatim}

Example Cont.
We should be able to check that the values are correct using the formula before... so we get:

\begin{verbatim}
pc1[1]
x[1,1]*coef[1] +x[1,2]*coef[2]
+x[1,3]*coef[3]
\end{verbatim}

Are they all off?!?

Some Matrix Background
The dot product of two vectors \( a \) and \( b \) is \( a_1b_1+a_2b_2+\cdots+a_qb_q \)
In vector notation this is:
It relates to distance by:
It relates to orthogonality by:
The Next Principal Component
The coefficient vector should be length 1.
The coefficient vector should be orthogonal to the previous one(s).
It should explain the largest possible amount of variance.

Eigenvalues and Eigenvectors
The eigenvectors of the covariance matrix are exactly the coefficients for principal components…
And the eigenvalues are the variances of the new variables!

Eigenvalues and Eigenvectors?!?
Consider the equation:
\[(\Sigma - \lambda I)x = 0\]
\(\lambda\) is an eigenvalue
\(x\) is an eigenvector
We typically make the length of \(x\) be 1.
We typically order them from largest \(\lambda\) to smallest.
Example Cont.
\[\text{eigen(cov(x))}$vectors\]
\[\text{princomp(x,cor=F)}$loadings\]
\[\text{eigen(cov(x))}$values\]
\[\text{round(var(princomp(x,cor=F)}$scores),2)\]

Scree Plot
\[\text{plot(princomp(x,cor=F))}\]

In Matrix Notation
\[Y=a_1X_1+a_2X_2+\cdots+a_qX_q\]
\[\text{or}\]
\[Y_{(n \times q)}=X_{(n \times q)}A_{(q \times q)}\]
Reversing It

If we have $Y$ we should be able to get $X$... say using regression.

In fact in this case we should get it exactly!

$$Y_{(nxq)} A^{-1}_{(qxq)} = X_{(nxq)} A_{(qxq)} A^{-1}_{(qxq)}$$

$$X_{(nxq)} = Y_{(nxq)} A^{-1}_{(qxq)}$$

Checking What Was Lost

```r
sect3<-as.matrix(oildata[,12:31])
a<-eigen(cov(sect3))$vectors
y<-sect3%*%a
summary(lm(sect3[,1]~y))$r.squared
```}

What if we only kept 4 PC’s?

```r
y4<-y[,1:4]
ssect4<-sect3
for (i in 1:20){
  sect3est[,i]<-
    lm(sect3[,i]~y4)$fitted.values
}
```
Lots of Lost Information!

s3var<-apply(sect3,2,var)
s3varest<-apply(sect3est,2,var)
cbind(s3var,s3varest)
cor(sect3)[1:4,1:4]
cor(sect3est)[1:4,1:4]